	Type	Hits	Search Text	Time Stamp	Ref #
1	IS&R	1930	(707/200).CCLS.	2005/05/15 09:24	S79
2	IS&R	1111	(707/204).CCLS.	2005/05/15 09:24	S80
3	IS&R	791	(707/205).CCLS.	2005/05/15 09:24	S81
4	IS&R	278	(711/138).CCLS.	2005/05/15 09:24	S82
5	IS&R	969	(710/22).CCLS.	2005/05/15 09:24	S83
6	IS&R	527	(707/206).CCLS.	2005/05/15 09:23	S78
7	BRS	3	S75 and S76	2005/05/15 09:22	S77
8	BRS	401	cache same garbage	2005/05/15 09:09	S76
9	BRS	494	temporal with cache	2005/05/15 09:08	S71
10	BRS	136	temporal with streaming	2005/05/15 09:08	S75
11	BRS	357	pollut$5 with cache	2005/05/15 09:06	S72
12	BRS	57	S71 and S72	2005/05/15 09:06	S73
13	BRS	2	S73 and garbage	2005/05/15 09:06	S74
14	BRS	255	garbage same cache	2005/05/15 09:05	S1
15	BRS	21	S68 and garbage	2005/05/06 18:55	S69
16	BRS	14	S69 and temporal	2005/05/06 18:55	S70

	DBs	Comments	Error Definition	Errors
1	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB			
2	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB			
3	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB			
4	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB			
5	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB			
6	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB			
7	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB			
8	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB			
9	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB			
10	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB			
11	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB			
12	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB			
13	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB			
14	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB			
15	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB			
16	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB			

ProQuest®

Results – powered by ProQuest® Smart Search

Suggested Topics About < Previous | Next >

Cache
Cache AND Internet

Browse Suggested Publications About < Previous | Next >

M2 Presswire; Coventry
Journal of Consumer Research; Gainesville

2 documents found for: *non-temporal and cache* Set up Alert About

Dissertations

Mark / Clear all on page | View marked documents | Show only full text | Sort results by: Most recent first

1. **Caching strategies for more efficient generational garbage collection**
by *Krishnakumar, Sree Vidhya Lakshmi*, M.S., **North Carolina State University**, 2004, 54 pages; AAT EP10016
Abstract

2. **Performance aspects of high-bandwidth multi-lateral cache organizations**
by *Rivers, Jude A.*, Ph.D., **University of Michigan**, 1998, 186 pages; AAT 9825335
Abstract | 24 Page Preview | Page Image - PDF

1-2 of 2

Want an alert for new results sent by email? Set up Alert About

Results per page: 30

Basic Search

Tools: Search Tips | Browse Topics | 2 Recent Searches

non-temporal and cache Search Clear

Database: Multiple databases... Select multiple databases

Date range: All dates

Limit results to: Full text documents only

Scholarly journals, including peer-reviewed About

More Search Options

Terms used
non temporal stores garbage collection cache pollute

Found **41** of **35,276** searched out of **53,499.**

Sort results by: relevance
Display results: expanded form
Save results to a Binder
Search Tips
Open results in a new window

Try an Advanced Search
Try this search in The ACM Guide

Results 1 - 20 of 41 Result page: **1** 2 3 next

Relevance scale

1 Concurrent garbage collection using hardware-assisted profiling
Timothy H. Heil, James E. Smith
October 2000 **ACM SIGPLAN Notices , Proceedings of the 2nd international symposium on Memory management**, Volume 36 Issue 1
Full text available: pdf(1.74 MB) Additional Information: full citation, abstract, citings, index terms

In the presence of on-chip multithreading, a Virtual Machine (VM) implementation can readily take advantage of *service threads* for enhancing performance by performing tasks such as profile collection and analysis, dynamic optimization, and garbage collection concurrently with program execution. In this context, a hardware-assisted profiling mechanism is proposed. The *Relational Profiling Architecture* (RPA) is designed from the top down. RPA is based on a relational model similar ...

2 Software prefetching for mark-sweep garbage collection: hardware analysis and software redesign
Chen-Yong Cher, Antony L. Hosking, T. N. Vijaykumar
October 2004 **Proceedings of the 11th international conference on Architectural support for programming languages and operating systems**, Volume 38 , 39 , 32 Issue 5 , 11 , 5
Full text available: pdf(165.32 KB) Additional Information: full citation, abstract, references, index terms

Tracing garbage collectors traverse references from live program variables, transitively tracing out the closure of live objects. Memory accesses incurred during tracing are essentially random: a given object may contain references to any other object. Since application heaps are typically much larger than hardware caches, tracing results in many cache misses. Technology trends will make cache misses more important, so tracing is a prime target for prefetching.Simulation of Java benchmarks runni ...

Keywords: breadth-first, buffered prefetch, cache architecture, depth-first, garbage collection, mark-sweep, prefetch-on-grey, prefetching

3 The case for profile-directed selection of garbage collectors
Robert Fitzgerald, David Tarditi
October 2000 **ACM SIGPLAN Notices , Proceedings of the 2nd international symposium on Memory management**, Volume 36 Issue 1
Full text available: pdf(1.28 MB) Additional Information: full citation, abstract, citings, index terms

Many garbage-collected systems use a single garbage collection algorithm across all applications. It has long been known that this can produce poor performance on applications for which that collector is not well suited. In some systems, such as those that execute stand-alone compiled executables, an appropriate collector for each application can be

selected from a pool of available collectors and tuned by using profile information. In a study of 20 benchmarks and several collectors, compiled ...

4 Contaminated garbage collection

Dante J. Cannarozzi, Michael P. Plezbert, Ron K. Cytron

May 2000 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 2000 conference on Programming language design and implementation**, Volume 35 Issue 5

Full text available: pdf(559.20 KB) Additional Information: full citation, abstract, references, citings, index terms

We describe a new method for determining when an object can be garbage collected. The method does not require marking live objects. Instead, each object X is dynamically associated with a stack frame M, such that Xis collectable when M pops. Because X could have been dead earlier, our method is conservative. Our results demonstrate that the method nonetheless identifies a large percentag ...

5 Memory forwarding: enabling aggressive layout optimizations by guaranteeing the safety of data relocation

Chi-Keung Luk, Todd C. Mowry

May 1999 **ACM SIGARCH Computer Architecture News , Proceedings of the 26th annual international symposium on Computer architecture**, Volume 27 Issue 2

Full text available: pdf(196.77 KB) Additional Information: full citation, abstract, references, citings, index terms

Publisher Site

By optimizing data layout at run-time, we can potentially enhance the performance of caches by actively creating spatial locality, facilitating prefetching, and avoiding cache conflicts and false sharing. Unfortunately, it is extremely difficult to guarantee that such optimizations are *safe* in practice on today's machines, since accurately updating *all* pointers to an object requires perfect alias information, which is well beyond the scope of the compiler for languages such as C. T ...

6 The cache behaviour of large lazy functional programs on stock hardware

Nicholas Nethercote, Alan Mycroft

June 2002 **ACM SIGPLAN Notices , Proceedings of the workshop on Memory system performance**, Volume 38 Issue 2 supplement

Full text available: pdf(1.26 MB) Additional Information: full citation, abstract, references, index terms

Lazy functional programs behave differently from imperative programs and these differences extend to cache behaviour. We use hardware counters and a simple yet accurate execution cost model to analyse some large Haskell programs on the x86 architecture. The programs do not interact well with modern processors---L2 cache data miss stalls and branch misprediction stalls account for up to 60% and 32% of execution time respectively. Moreover, the program code exhibits little exploitable instruction- ...

Keywords: Glasgow Haskell Compiler, Haskell, branch misprediction, cache measurement, cache simulation, hardware counters

7 Improving the cache locality of memory allocation

Dirk Grunwald, Benjamin Zorn, Robert Henderson

June 1993 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 1993 conference on Programming language design and implementation**, Volume 28 Issue 6

Full text available: pdf(1.20 MB) Additional Information: full citation, abstract, references, citings, index terms

The allocation and disposal of memory is a ubiquitous operation in most programs. Rarely do programmers concern themselves with details of memory allocators; most assume that memory allocators provided by the system perform well. This paper presents a performance evaluation of the reference locality of dynamic storage allocation algorithms based on trace-driven simualtion of five large allocation-intensive C programs. In this paper, we show how the design of a memory allocator can significa ...

8 Concurrency: Write barrier elision for concurrent garbage collectors
Martin T. Vechev, David F. Bacon
October 2004 **Proceedings of the 4th international symposium on Memory management**
Full text available: pdf(490.73 KB) Additional Information: full citation, abstract, references, index terms

Concurrent garbage collectors require write barriers to preserve consistency, but these barriers impose significant direct and indirect costs. While there has been a lot of work on optimizing write barriers, we present the first study of their elision in a concurrent collector. We show conditions under which write barriers are redundant, and describe how these conditions can be applied to both incremental update or snapshot-at-the-beginning barriers. We then evaluate the potential for write b ...

Keywords: concurrent garbage collection, write barrier

9 Space efficient conservative garbage collection
Hans-Juergen Boehm
June 1993 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 1993 conference on Programming language design and implementation**, Volume 28 Issue 6
Full text available: pdf(1.03 MB) Additional Information: full citation, abstract, references, citings, index terms

We call a garbage collector conservative if it has only partial information about the location of pointers, and is thus forced to treat arbitrary bit patterns as though they might be pointers, in at least some cases. We show that some very inexpensive, but previously unused techniques can have dramatic impact on the effectiveness of conservative garbage collectors in reclaiming memory. Our most significant observation is that static data that appears to point to the heap should not result i ...

10 Exploiting prolific types for memory management and optimizations
Yefim Shuf, Manish Gupta, Rajesh Bordawekar, Jaswinder Pal Singh
January 2002 **ACM SIGPLAN Notices , Proceedings of the 29th ACM SIGPLAN-SIGACT symposium on Principles of programming languages**, Volume 37 Issue 1
Full text available: pdf(203.59 KB) Additional Information: full citation, abstract, references, citings

In this paper, we introduce the notion of *prolific* and *non-prolific* types, based on the number of instantiated objects of those types. We demonstrate that distinguishing between these types enables a new class of techniques for memory management and data locality, and facilitates the deployment of known techniques. Specifically, we first present a new *type-based* approach to garbage collection that has similar attributes but lower cost than generational collection. Then we de ...

11 Prefetch injection based on hardware monitoring and object metadata
Ali-Reza Adl-Tabatabai, Richard L. Hudson, Mauricio J. Serrano, Sreenivas Subramoney
June 2004 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 2004 conference on Programming language design and implementation**, Volume 39 Issue 6
Full text available: pdf(288.00 KB) Additional Information: full citation, abstract, references, index terms

Cache miss stalls hurt performance because of the large gap between memory and processor speeds - for example, the popular server benchmark SPEC JBB2000 spends 45% of its cycles stalled waiting for memory requests on the Itanium® 2 processor. Traversing linked data structures causes a large portion of these stalls. Prefetching for linked data structures remains a major challenge because serial data dependencies between elements in a linked data structure preclude the timely materialization ...

Keywords: cache misses, compiler optimization, garbage collection, prefetching, profile-guided optimization, virtual machines

12

Escape analysis for Java™: Theory and practice
Bruno Blanchet
November 2003 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 25 Issue 6
Full text available: pdf(684.21 KB) Additional Information: full citation, abstract, references, citings, index terms, review

Escape analysis is a static analysis that determines whether the lifetime of data may exceed its static scope.This paper first presents the design and correctness proof of an escape analysis for Java™. This analysis is interprocedural, context sensitive, and as flow-sensitive as the static single assignment form. So, assignments to object fields are analyzed in a flow-insensitive manner. Since Java is an imperative language, the effect of assignments must be precisely determined. Thi ...

Keywords: Java, optimization, stack allocation, static analysis, synchronization elimination

13 A stateless, content-directed data prefetching mechanism
Robert Cooksey, Stephan Jourdan, Dirk Grunwald
October 2002 **Proceedings of the 10th international conference on Architectural support for programming languages and operating systems**, Volume 37 , 30 , 36 Issue 10 , 5 , 5
Full text available: pdf(1.20 MB) Additional Information: full citation, abstract, references, citings

Although central processor speeds continues to improve, improvements in overall system performance are increasingly hampered by memory latency, especially for pointer-intensive applications. To counter this loss of performance, numerous data and instruction prefetch mechanisms have been proposed. Recently, several proposals have posited a *memory-side* prefetcher; typically, these prefetchers involve a distinct processor that executes a program slice that would effectively prefetch data nee ...

14 Static load classification for improving the value predictability of data-cache misses
Martin Burtscher, Amer Diwan, Matthias Hauswirth
May 2002 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 2002 Conference on Programming language design and implementation**, Volume 37 Issue 5
Full text available: pdf(273.26 KB) Additional Information: full citation, abstract, references, index terms

While caches are effective at avoiding most main-memory accesses, the few remaining memory references are still expensive. Even one cache miss per one hundred accesses can double a program's execution time. To better tolerate the data-cache miss latency, architects have proposed various speculation mechanisms, including load-value prediction. A load-value predictor guesses the result of a load so that the dependent instructions can immediately proceed without having to wait for the memory access ...

Keywords: load-value prediction, type-based analysis

15 A fully associative software-managed cache design
Erik G. Hallnor, Steven K. Reinhardt
May 2000 **ACM SIGARCH Computer Architecture News , Proceedings of the 27th annual international symposium on Computer architecture**, Volume 28 Issue 2
Full text available: pdf(117.18 KB) Additional Information: full citation, abstract, references, citings, index terms

As DRAM access latencies approach a thousand instruction-execution times and on-chip caches grow to multiple megabytes, it is not clear that conventional cache structures continue to be appropriate. Two key features—full associativity and software management—have been used successfully in the virtual-memory domain to cope with disk access latencies. Future systems will need to employ similar techniques to deal with DRAM latencies. This paper presents a practical, fully associati ...

16

Reconciling responsiveness with performance in pure object-oriented languages
Urs Hölzle, David Ungar
July 1996 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 18 Issue 4

Full text available: pdf(537.19 KB) Additional Information: full citation, abstract, references, citings, index terms, review

Dynamically dispatched calls often limit the performance of object-oriented programs, since opject-oriented programming encourages factoring code into small, reusable units, thereby increasing the frequency of these expensive operations. Frequent calls not only slow down execution with the dispatch overhead per se, but more importantly they hinder optimization by limiting the range and effectiveness of standard global optimizations. In particular, dynamically dispatched calles prevent stand ...

Keywords: adaptive optimization, pause clustering, profile-based optimization, run-time compilation, type feedback

17 Dynamic hot data stream prefetching for general-purpose programs
Trishul M. Chilimbi, Martin Hirzel
May 2002 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 2002 Conference on Programming language design and implementation**, Volume 37 Issue 5

Full text available: pdf(210.85 KB) Additional Information: full citation, abstract, references, citings, index terms

Prefetching data ahead of use has the potential to tolerate the growing processor-memory performance gap by overlapping long latency memory accesses with useful computation. While sophisticated prefetching techniques have been automated for limited domains, such as scientific codes that access dense arrays in loop nests, a similar level of success has eluded general-purpose programs, especially pointer-chasing codes written in languages such as C and C++. We address this problem by describing ...

Keywords: data reference profiling, dynamic optimization, dynamic profiling, memory performance optimization, prefetching, temporal profiling

18 Effective jump-pointer prefetching for linked data structures
Amir Roth, Gurindar S. Sohi
May 1999 **ACM SIGARCH Computer Architecture News , Proceedings of the 26th annual international symposium on Computer architecture**, Volume 27 Issue 2

Full text available: pdf(113.33 KB) Publisher Site Additional Information: full citation, abstract, references, citings, index terms

Current techniques for prefetching linked data structures (LDS) exploit the work available in one loop iteration or recursive call to overlap pointer chasing latency. Jump pointers, which provide direct access to non-adjacent nodes, can be used for prefetching when loop and recursive procedure bodies are small and do not have sufficient work to overlap a long latency. This paper describes a framework for jump-pointer prefetching (JPP) that supports four prefetching idioms: queue, full, chain, an ...

19 Using lifetime predictors to improve memory allocation performance
David A. Barrett, Benjamin G. Zorn
June 1993 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 1993 conference on Programming language design and implementation**, Volume 28 Issue 6

Full text available: pdf(1.08 MB) Additional Information: full citation, abstract, references, citings, index terms

Dynamic storage allocation is used heavily in many application areas including interpreters, simulators, optimizers, and translators. We describe research that can improve all aspects of the performance of dynamic storage allocation by predicting the lifetimes of short-lived objects when they are allocated. Using five significant, allocation-intensive C programs, we

show that a great fraction of all bytes allocated are short-lived (> 90% in all cases). Furthermore, we describe an algorit ...

20 I/O reference behavior of production database workloads and the TPC benchmarks—an analysis at the logical level

Windsor W. Hsu, Alan Jay Smith, Honesty C. Young

March 2001 **ACM Transactions on Database Systems (TODS)**, Volume 26 Issue 1

Full text available: pdf(5.42 MB) Additional Information: full citation, abstract, references, citings, index terms

As improvements in processor performance continue to far outpace improvements in storage performance, I/O is increasingly the bottleneck in computer systems, especially in large database systems that manage huge amoungs of data. The key to achieving good I/O performance is to thoroughly understand its characteristics. In this article we present a comprehensive analysis of the logical I/O reference behavior of the peak productiondatabase workloads from ten of the world's largest corporatio ...

Keywords: I/O, TPC benchmarks, caching, locality, prefetching, production database workloads, reference behavior, sequentiality, workload characterization

Results 1 - 20 of 41 Result page: **1** 2 3 next

Terms used
non temporal stores garbage collection cache pollute

Found **41** of **53,499**

Sort results by: relevance
Display results: expanded form
Save results to a Binder
Search Tips
Open results in a new window

Try an Advanced Search
Try this search in The ACM Guide

Results 21 - 40 of 41 Result page: previous 1 **2** 3

Relevance scale

21 Decentralized storage systems: Farsite: federated, available, and reliable storage for an incompletely trusted environment

Atul Adya, William J. Bolosky, Miguel Castro, Gerald Cermak, Ronnie Chaiken, John R. Douceur, Jon Howell, Jacob R. Lorch, Marvin Theimer, Roger P. Wattenhofer
December 2002 **ACM SIGOPS Operating Systems Review**, Volume 36 Issue SI

Full text available: pdf(1.87 MB) Additional Information: full citation, abstract, references

Farsite is a secure, scalable file system that logically functions as a centralized file server but is physically distributed among a set of untrusted computers. Farsite provides file availability and reliability through randomized replicated storage; it ensures the secrecy of file contents with cryptographic techniques; it maintains the integrity of file and directory data with a Byzantine-fault-tolerant protocol; it is designed to be scalable by using a distributed hint mechanism and delegatio ...

22 Trap-driven memory simulation with Tapeworm II

Richard Uhlig, David Nagle, Trevor Mudge, Stuart Sechrest
January 1997 **ACM Transactions on Modeling and Computer Simulation (TOMACS)**, Volume 7 Issue 1

Full text available: pdf(630.91 KB) Additional Information: full citation, references, index terms

Keywords: Cache, TLB, memory system, simulation, trace-driven simulation, trap-driven simulation

23 Functioning without closure: type-safe customized function representations for standard ML

Allyn Dimock, Ian Westmacott, Robert Muller, Franklyn Turbak, J. B. Wells
October 2001 **ACM SIGPLAN Notices , Proceedings of the sixth ACM SIGPLAN international conference on Functional programming**, Volume 36 Issue 10

Full text available: pdf(257.35 KB) Additional Information: full citation, abstract, references, citings, index terms

The CIL compiler for core Standard ML compiles whole ML programs using a novel typed intermediate language that supports the generation of type-safe customized data representations. In this paper, we present empirical data comparing the relative efficacy of several different flow-based customization strategies for function representations. We develop a cost model to interpret dynamic counts of operations required for each strategy. In this cost model, customizing the representation of closed fun ...

24 Replacement policies for a proxy cache
Luigi Rizzo, Lorenzo Vicisano
April 2000 **IEEE/ACM Transactions on Networking (TON)**, Volume 8 Issue 2
Full text available: pdf(277.42 KB) Additional Information: full citation, references, citings, index terms

Keywords: Web, caching, communication networks, policies, replacement

25 Departments: Internet nuggets
Mark Thorson
September 2003 **ACM SIGARCH Computer Architecture News**, Volume 31 Issue 4
Full text available: pdf(708.57 KB) Additional Information: full citation

26 New techniques for security and reliability enhancement in embedded systems: Analyzing heap error behavior in embedded JVM environments
G. Chen, M. Kandemir, N. Vijaykrishnan, A. Sivasubramaniam, M. J. Irwin
September 2004 **Proceedings of the 2nd IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis**
Full text available: pdf(285.76 KB) Additional Information: full citation, abstract, references, index terms

Recent studies have shown that transient hardware errors caused by external factors such as alpha particles and cosmic ray strikes can be responsible for a large percentage of system down-time. Denser processing technologies, increasing clock speeds, and low supply voltages used in embedded systems can worsen this problem. In many embedded environments, one may not want to provision extensive error protection in hardware because of (i) form-factor or power consumption limitations, and/or (ii) to ...

Keywords: JVM, softerrors

27 Object combining: A new aggressive optimization for object intensive programs
Ronald Veldema, J. H. Ceriel, F. H. Rutger, E. Henri
November 2002 **Proceedings of the 2002 joint ACM-ISCOPE conference on Java Grande**
Full text available: pdf(99.27 KB) Additional Information: full citation, abstract, references, index terms

Object combining tries to put objects together that have roughly the same life times in order to reduce strain on the memory manager and to reduce the number of pointer indirections during a program's execution. Object combining works by appending the fields of one object to another, allowing allocation and freeing of multiple objects with a single heap (de) allocation. Unlike object *inlining*, which will only optimize objects where one has a (unique) pointer to another, our optimization al ...

Keywords: Java, garbage collection, object management

28 Jedd: a BDD-based relational extension of Java
Ondřej Lhoták, Laurie Hendren
June 2004 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 2004 conference on Programming language design and implementation**, Volume 39 Issue 6
Full text available: pdf(137.26 KB) Additional Information: full citation, abstract, references, citings, index terms

In this paper we present Jedd, a language extension to Java that supports a convenient way of programming with Binary Decision Diagrams (BDDs). The Jedd language abstracts BDDs as database-style relations and operations on relations, and provides static type rules to ensure that relational operations are used correctly.The paper provides a description of the

Jedd language and reports on the design and implementation of the Jedd translator and associated runtime system. Of particular interest is ...

Keywords: Java, binary decision diagrams, boolean formula satisfiability, language design, program analysis, relations

29 Dependence based prefetching for linked data structures
Amir Roth, Andreas Moshovos, Gurindar S. Sohi
October 1998 **Proceedings of the eighth international conference on Architectural support for programming languages and operating systems**, Volume 32 , 33 Issue 5 , 11

Full text available: pdf(1.81 MB) Additional Information: full citation, abstract, references, citings, index terms

We introduce a dynamic scheme that captures the accesspat-terns of linked data structures and can be used to predict future accesses with high accuracy. Our technique exploits the dependence relationships that exist between loads that produce addresses and loads that consume these addresses. By identzj+ing producer-consumer pairs, we construct a compact internal representation for the associated structure and its traversal. To achieve a prefetching eflect, a small prefetch engine speculatively t ...

30 Making a fast curry: push/enter vs. eval/apply for higher-order languages
Simon Marlow, Simon Peyton Jones
September 2004 **ACM SIGPLAN Notices , Proceedings of the ninth ACM SIGPLAN international conference on Functional programming**, Volume 39 Issue 9

Full text available: pdf(186.19 KB) Additional Information: full citation, abstract, references, citings, index terms

Higher-order languages that encourage currying are implemented using one of two basic evaluation models: push/enter or eval/apply. Implementors use their intuition and qualitative judgements to choose one model or the other.Our goal in this paper is to provide, for the first time, a more substantial basis for this choice, based on our qualitative and quantitative experience of implementing both models in a state-of-the-art compiler for Haskell.Our conclusion is simple, and contradicts our initia ...

31 Optimizing dynamically-dispatched calls with run-time type feedback
Urs Hölzle, David Ungar
June 1994 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 1994 conference on Programming language design and implementation**, Volume 29 Issue 6

Full text available: pdf(1.39 MB) Additional Information: full citation, references, citings, index terms, review

32 The VMP multiprocessor: initial experience, refinements, and performance evaluation
D. R. Cheriton, A. Gupta, P. D. Boyle, H. A. Goosen
May 1988 **ACM SIGARCH Computer Architecture News , Proceedings of the 15th Annual International Symposium on Computer architecture**, Volume 16 Issue 2

Full text available: pdf(1.73 MB) Additional Information: full citation, abstract, references, citings, index terms

VMP is an experimental multiprocessor being developed at Stanford University, suitable for high-performance workstations and server machines. Its primary novelty lies in the use of software management of the per-processor caches and the design decisions in the cache and bus that make this approach feasible. The design and some uniprocessor trace-driven simulations indicating its performance have been reported previously. In this paper, we present our initial experience with the V ...

33 The use of multithreading for exception handling
Craig B. Zilles, Joel S. Emer, Gurindar S. Sohi
November 1999 **Proceedings of the 32nd annual ACM/IEEE international symposium on**

Microarchitecture

Full text available: pdf(1.49 MB) Publisher Site Additional Information: full citation, abstract, references, citings, index terms

Common hardware exceptions, when implemented by trapping, unnecessarily serialize program execution in dynamically scheduled superscalar processors. To avoid the consequences of trapping the main program thread, multithreaded CPUs can exploit control and data independence by executing the exception handler in a separate hardware context. The main thread doesn't squash instructions after the excepting instruction, conserving fetch bandwidth and allowing execution of instructions inde ...

34 Exploiting ILP in page-based intelligent memory

Mark Oskin, Justin Hensley, Diana Keen, Frederic T. Chong, Matthew Farrens, Aneet Chopra

November 1999 **Proceedings of the 32nd annual ACM/IEEE international symposium on Microarchitecture**

Full text available: pdf(1.35 MB) Publisher Site Additional Information: full citation, abstract, references, index terms

This study compares the speed, area, and power of different implementations of Active Pages [OCS98], an intelligent memory system which helps bridge the growing gap between processor and memory performance by associating simple functions with each page of data. Previous investigations have shown up to 1000X speedups using a block of reconfigurable logic to implement these functions next to each sub-array on a DRAM chip. In this study, we show that instruction-level parallelism, n ...

35 Efficient shared memory with minimal hardware support

Leonidas I. Kontothanassis, Michael L. Scott

September 1995 **ACM SIGARCH Computer Architecture News**, Volume 23 Issue 4

Full text available: pdf(536.07 KB) Additional Information: full citation, abstract, index terms

Shared memory is widely regarded as a more intuitive model than message passing for the development of parallel programs. A shared memory model can be provided by hardware, software, or some combination of both. One of the most important problems to be solved in shared memory environments is that of cache coherence. Experience indicates, unsurprisingly, that hardware-coherent multiprocessors greatly outperform distributed shared-memory (DSM) emulations on message-passing hardware. Intermediate o ...

36 Memory hierarchy: Polynomial-time algorithm for on-chip scratchpad memory partitioning

Federico Angiolini, Luca Benini, Alberto Caprara

October 2003 **Proceedings of the 2003 international conference on Compilers, architecture and synthesis for embedded systems**

Full text available: pdf(215.57 KB) Additional Information: full citation, abstract, references, citings, index terms

Focusing on embedded applications, scratchpad memories (SPMs) look like a best-compromise solution when taking into account performance, energy consumption and die area. The main challenge in SPM design is mapping memory locations to scratchpad locations. This paper describes an algorithm to optimally solve such a mapping problem by means of Dynamic Programming applied to a synthesizable hardware architecture. The algorithm works by mapping segments of external memory to physically partitioned b ...

Keywords: design automation, dynamic programming, embedded design, memory hierarchy, partitioning algorithm, power saving, scratchpad memory

37 The Web Service Discovery Architecture

Wolfgang Hoschek

November 2002 **Proceedings of the 2002 ACM/IEEE conference on Supercomputing**

Full text available: pdf(282.28 KB) Additional Information: full citation, abstract, references, index terms

In this paper, we propose the Web Service Discovery Architecture (WSDA). At runtime, Grid applications can use this architecture to discover and adapt to remote services. WSDA promotes an interoperable web service discovery layer by defining appropriate services, interfaces, operations and protocol bindings, based on industry standards. It is unified because it subsumes an array of disparate concepts, interfaces and protocols under a single semi-transparent umbrella. It is modular because it def ...

38 Transparent proxies for java futures

Polyvios Pratikakis, Jaime Spacco, Michael Hicks

October 2004 **ACM SIGPLAN Notices , Proceedings of the 19th annual ACM SIGPLAN Conference on Object-oriented programming, systems, languages, and applications**, Volume 39 Issue 10

Full text available: pdf(351.78 KB) Additional Information: full citation, abstract, references, index terms

A proxy object is a surrogate or placeholder that controls access to another target object. Proxies can be used to support distributed programming, lazy or parallel evaluation, access control, and other simple forms of behavioral reflection. However, wrapper proxies (like futures or suspensions for yet-to-be-computed results) can require significant code changes to be used in statically-typed languages, while proxies more generally c ...

Keywords: Java, future, proxy, type inference, type qualifier

39 AVANCE: an object management system

Anders Bjornerstedt, Stefan Britts

January 1988 **ACM SIGPLAN Notices , Conference proceedings on Object-oriented programming systems, languages and applications**, Volume 23 Issue 11

Full text available: pdf(1.87 MB) Additional Information: full citation, abstract, references, citings, index terms

AVANCE1 is an integrated application development and run-time system. It provides facilities for programming with shared and persistent objects, transactions and processes. The architecture is designed with decentralization in mind by having a large object identifier space and a remote procedure call interface to objects. Emphasis in this paper is on the programming language PAL and its relation with the underlying virtual machine.

40 Abstracting remote object interaction in a peer-2-peer environment

Patrick Thomas Eugster, Sebastien Baehni

November 2002 **Proceedings of the 2002 joint ACM-ISCOPE conference on Java Grande**

Full text available: pdf(202.02 KB) Additional Information: full citation, abstract, references, index terms

Leveraged by the success of applications aiming at the "free" sharing of data in the Internet, the paradigm of peer-to-peer (P2P) computing has been devoted substantial consideration recently.This paper presents an abstraction for remote object interaction in a P2P environment, called borrow/lend (BL). We present the principles underlying our BL abstraction, and its implementation in Java. We contrast our abstraction with established abstractions for distributed programming such as the remote me ...

Keywords: Java, abstraction, borrow/lend, peer-to-peer, type

Terms used
non temporal stores garbage collection cache pollute

Found **41** of **53,499**

Sort results by: relevance
Display results: expanded form
Save results to a Binder
Search Tips
Open results in a new window

Try an Advanced Search
Try this search in The ACM Guide

Results 41 - 41 of 41 Result page: previous 1 2 **3**

Relevance scale

41 Configuration cloning: exploiting regularity in dynamic DSP architectures
S. R. Park, W. Burleson
February 1999 **Proceedings of the 1999 ACM/SIGDA seventh international symposium on Field programmable gate arrays**
Full text available: pdf(1.72 MB) Additional Information: full citation, references, citings, index terms

Results 41 - 41 of 41 Result page: previous 1 2 **3**